In the Claims:

1. (Twice amended) A method for performing deep packet processing on an input variable word bit chain, said method comprising the steps of:

- creating a state table based on at least one initial state, and a final state, said ~~each~~ state table having entries, each said entry defining a state-transition rule comprising a s-bit current state, a n-bit word of the input variable word bit chain and a s-bit ~~as-bit~~ next state;

- converting the entries of the state table into a reduced number of state-transition rule entries, each entry containing a ternary match condition expressed as a s+n-bit test value and a s+n-bit test mask to be applied to a ~~the~~ current state and an ~~the~~ input word in combination, said each entry further containing the s-bit next state, to provide a reduced state table with entries;

- ordering the reduced state table entries obtained by the execution of the converting ~~preceding~~ step, in a prioritized order, with most frequently used state-transition rules having the highest priority;

-initializing a current state as being the initial state and a ~~the~~ first word of the chain being a current input word;

- testing a ~~the~~ current state and a ~~the~~ current input in combination, against the test value, using the test mask, in all the entries of the reduced state table until a match is found on at least one entry;

- if multiple entries match, selecting one entry with the highest priority to be a selected matching entry;

- if the next state read in the state-transition rule of the selected matching entry is not a final state, defining a ~~the~~ next word of the input word chain as being a ~~the~~ current input and the next state being the current state;

- repeating the testing, selecting and defining steps until a final state is found, and before the initializing step comprising the further steps of:

- defining as a hash index for the reduced state table, a set of i bit locations inside the s-bit current state and an ~~the~~ input n-bit word in combination, and an integer N, such that, at most, N table entries can match a hash index value;

- creating a compressed state table, indexed by the hash index, having 2^i entries, each entry corresponding to one value of the hash index, and each having a maximum of N state-transition rules of the reduced state table corresponding to the same hash index value and written in a priority order;

- saving an s + n bit index mask corresponding to the hash index, and saving a base address pointer of the compressed state table; and

said testing step further comprises an initial step of identifying the hash index of a ~~the~~ current state ~~sate~~ and current input in combination, using the index mask, and testing the hash index to identify a ~~the~~ corresponding entry in the compressed state table located using the base address pointer, the following testing step against the test value and the following steps being performed on the maximum of N state-transition rules of the identified said corresponding entry in the compressed state table ~~entry~~,

- dividing the compressed state table into more than one compressed state sub-table;

- extending in each of the compressed state sub-tables, each state-transition rule with a corresponding index mask and a base address pointer of the compressed state sub-table of the next state in said state-transition rule;

- initializing a current compressed state sub-table base address pointer, and

- the base address pointer of a matching entry becoming the current base address pointer of the compressed state of the next state.